NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CONTEXT-BASED MOBILE SECURITY ENCLAVE

by

Joey C. Carter

September 2012

Thesis Advisor: Gurminder Singh
Thesis Co-Advisor: John Gibson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2012	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Context-Based Mobile Security Enclave	5. FUNDING NUMBERS
6. AUTHOR(S) Joey C. Carter	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

Currently, there are no secure access control methods of controlling restricted material access on a mobile device using context-based authentication methods. Simple challenge/response protocols do not provide the security required for some restricted information. A lost/misplaced device or compromised password can easily lead to the compromise of any restricted information to which the device may have access. Due to the inherent portability of a mobile device, a broader, more comprehensive set of access controls is required.

This thesis will research and build a prototype application that will allow access to restricted information, stored on the device itself, based on a set of predefined contexts using the device's own hardware.

14. SUBJECT TERMS Android Programming, Security Application, Application Enclave	15. NUMBER OF PAGES 63
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

CONTEXT-BASED MOBILE SECURITY ENCLAVE

Joey C. Carter
Lieutenant, United States Navy
B.S., University of Arizona, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author: Joey C. Carter

Approved by: Gurminder Singh
Thesis Advisor

John Gibson
Thesis Co-Advisor

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Currently, there are no secure access control methods of controlling restricted material access on a mobile device using context-based authentication methods. Simple challenge/response protocols do not provide the security required for some restricted information. A lost/misplaced device or compromised password can easily lead to the compromise of any restricted information to which the device may have access. Due to the inherent portability of a mobile device, a broader, more comprehensive set of access controls is required.

This thesis will research and build a prototype application that will allow access to restricted information, stored on the device itself, based on a set of predefined contexts using the device's own hardware.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

A-GPS	Assisted Global Positioning System
ADB	Android Debugger
API	Application Programming Interface
APK	Android Application Package
BSC	Base Station Controller
BTS	Base Transceiver Station
BYOD	Bring Your Own Device
CDMA	Code Division Multiple Access
CID	Cellular Identification Number
DEX	Dalvik Executable
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HNI	Home Network Identity
IDE	Integrated Development Environment
IMEI	International Mobile Equipment Identifier
IMSI	International Mobile subscriber identity
IT	Information Technology
LAC	Location Area Code
LTE	Long-Term Evolution
MCC	Mobile Country Code
MNC	Mobile Network Code
MSIN	Mobile Station Identification Number
NANP	North American Numbering Plan
OEM	Original Equipment Manufacturer
OS	Operating System
PII	Personally Identifiable Information
PRN	Pseudorandom Number
SDK	Software Development Kit
SSL	Secured Socket Layer

SU	Super User
SIM	Subscriber Identity Module
VM	Virtual Machine

ACKNOWLEDGMENTS

Many thanks to my advisors, Dr. Gurminder Singh and Mr. John Gibson, for making this difficult experience as enjoyable and rewarding as possible. Without your extensive knowledge and guidance, this thesis would not have been possible.

I would also like to thank my classmates for all the assistance they have given me while navigating this challenging curriculum. I would especially like to thank Le Nolan for your unselfishness in devoting so much time in many of the topics our classmates and I struggled through.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

When a natural disaster occurs, within hours first responders are on the scene. They bring with them a plethora of equipment to help them aid the victims of the disaster. Most likely this equipment includes some sort of mobile device to access restricted personnel files/information. The security of the data and communications exchanged is paramount. The current mechanisms to secure that data restricting its access to a specified user in a specific situation or location are insufficient.

As mobile devices penetrate into the daily lives of personnel with access to restricted data, the risks of compromise of that data increase. Given how important mobile devices have become in the daily lives of their users, better ways of securing information on those devices are needed. According to the United States Department of Health and Human Services' Office for Civil Rights, there have been more than 420 security breaches in 2009 of the medical records of 19 million patients [1]. Many of these breaches involve critical personally identifiable information (PII) stored on portable electronic devices. The breaches are results of these devices falling into malicious hands without the proper security features to protect the data.

As businesses continually shift to a Bring Your Own Device (BYOD) policy, the history of security breaches since 2009 are evidence that the research into securing the information on those devices has not kept up with the pace

of that shift. Most data securing programs rely on a simple challenge/response protocol to protect the information, leaving all of the security in the hands of the user. A more reliable, administrator controlled set of security features is needed. An architecture designed around a mobile security enclave, with device-measured context access controls, addresses this increased security need.

A. MOBILE SECURITY ENCLAVE

A mobile security enclave is a method of access control between a collection of valuable assets and the mobile device. The enclave contains protected applications and only allows a user access to them when certain conditions exist [2]. These conditions, better known as contexts, are the key to the security of the applications. They must be out of the control of the user of the system and difficult to bypass. Hardware sensors of the mobile device can serve as an adequate set of security contexts to restrict access to protected applications.

B. OBJECTIVES

The goal of this thesis is to develop a prototype service that runs on a mobile device and restricts access to applications installed on the mobile device based on a set of contexts controlled by an administrator at setup. The prototype consists of an Android service (application) that utilizes built-in hardware to measure contexts on which to grant access to other protected (enclaved) applications and any data associated with them. In the initial service, three contexts will be used to grant

access to the enclave: global positioning system coordinates, base transceiver station identification (CID) and the international mobile subscriber identity (IMSI) number.

C. ORGANIZATION

Chapter I provides a brief discussion of the need for a more secure method of protecting proprietary/restricted information. The chapter is made up of two sections: One discusses the idea of the context-based application enclave and its associated contexts and the other explains the overall objectives of this thesis.

Chapter II provides a description of existing security programs and applications. The discussion includes their strengths and weaknesses and how the prototype Android service fills the gaps left by these applications. The Android operating system and included security features, along with the Android software development kit, are also discussed.

Chapter III outlines the architectural design used in creating the prototype Android service. The design of the contexts, enclave, and administrative functions are also discussed.

Chapter IV explains the implementation of the architectural design described in Chapter III. The use of the Android software development kit (SDK) and its associated classes are discussed. The administrator installation and setup are also included in this chapter, along with user functions and a step-by-step walkthrough.

Chapter V provides the reader with a summary and a brief overview of the prototype Android service created for this thesis. It is discussed in this chapter that it is possible to create an Android application that successfully protects other Android applications installed on the same device, using hardware-measured contexts. This chapter concludes with a discussion of possible future research related to this effort.

II. BACKGROUND

A. INTRODUCTION

This chapter provides the relevant background knowledge to support the following chapters. It includes descriptions of Bring Your Own Device (BYOD) policies, a discussion of Android Operating System inner-workings and security and related mobile device components and capabilities.

B. BRING YOUR OWN DEVICE

Since the advent of the smartphone, more wireless infrastructure in support of communications and data exchange is necessary to support the growing use of personal devices in the workplace. BYOD is a trend of employees using personally owned devices in their place of work for both personal and business use. While cost and convenience are the two dominant positive aspects of BYOD, there are a few negative aspects to a BYOD policy that, with good mitigation controls, can be substantially reduced.

A business that operates a BYOD policy shifts costs of data plans and devices to the user, can take advantage of newer technology faster, and have reduced IT support requirements. Allowing employees to use their personal devices for business transactions removes the need for both the employer to provide costly devices to the user and the need for the user to carry multiple devices. Users also tend to keep their devices on a faster replacement cycle than that of the average company, keeping electronic

infrastructure of the business more up-to-date. Since the company does not provide the devices, the need for IT support for the hardware is not required (support for software installed on the users' devices will most likely still be necessary, especially if organization-specific applications are deployed on the devices).

On the other-hand, the BYOD policy does not come without its risks. To start with, the content provider loses control over the IT hardware and how it is used. When businesses issue hardware to their employees, the devices are usually accompanied by usage policies, which dictate what can and can't be done with the device. Almost every aspect of the devices are managed and controlled by the management. With a BYOD policy, both business and personal applications/data are stored on the device. This leads to a higher possibility of compromise of restricted business applications/data by other users who may share the device or worse, malicious software designed with the intention of compromising said material. Clearly, loss or theft of the device affects data security whether the device of corporate-owned or BYOD.

While exercising a BYOD policy, a simple challenge/response protocol may not meet the restrictive needs to protect the information/data stored on the device. Compromise of the devices location, network connection, and user are not accounted for when utilizing a challenge/response protocol. A more comprehensive set of contexts on which to grant access is needed.

C. ANDROID OPERATING SYSTEM

The Android Operating System is built on the Linux kernel with libraries and Application Programming Interfaces (APIs), which use Java compatible libraries based on Apache Harmony (an open source Java implementation developed by the Apache Software Foundation). Android APIs are a set of programming code that allows applications to communicate with the operating system [3]. The set of APIs that are included in the Android Software Development Kit (SDK) is what allows the programmer to create applications on or for the Android OS that uses the device's hardware. Many of the APIs are open to the public and give access to services such as basic telephony data, location specific information (which is used in this thesis), and GSM and CDMA functions. Others are closed to the public and need special permissions for use, or are strictly controlled by the original equipment manufacturers (OEM) [3]. Figure 1 depicts the Android programming architecture as presented on the Android Developer's site [4].

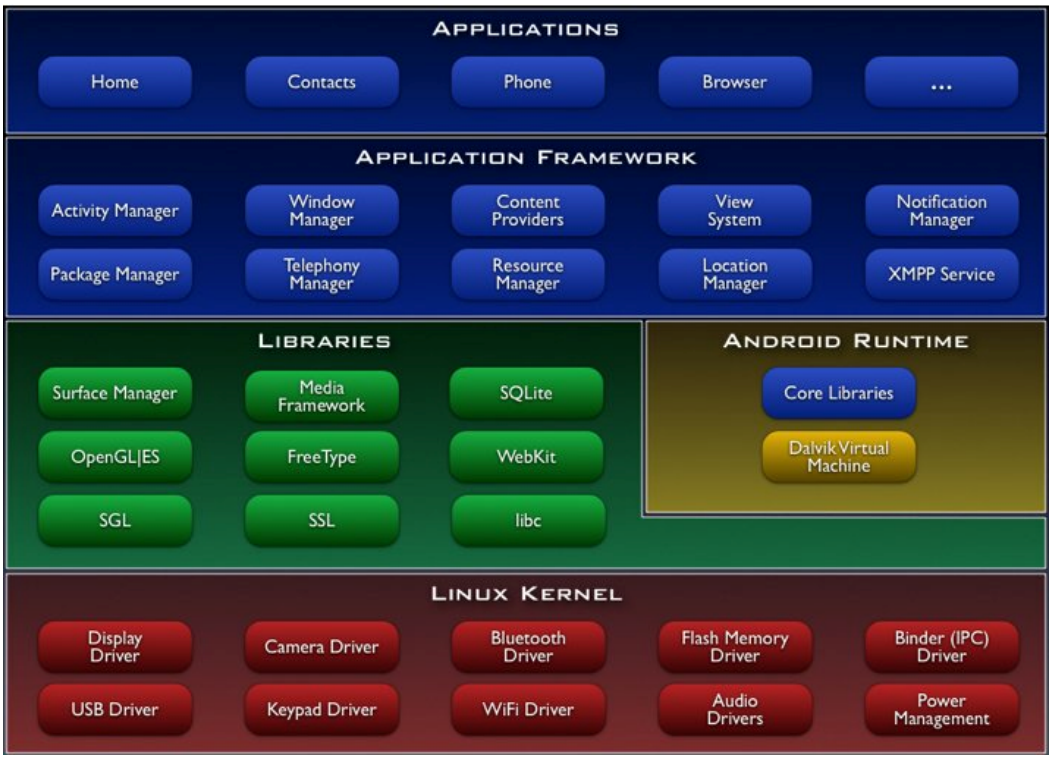

Figure 1. Android Application Architecture

The Android operating system uses the Dalvik virtual machine with “just-in-time” compilation to run Dalvik Executable (DEX) code. The Dalvik virtual machine is a register-based process virtual machine used to run applications in the Android Operating System. Designed by Google Engineer Dan Bornstein with contributions from other Google engineers, it minimizes memory requirements while at the same time allowing multiple VMs to run at the same time [5]. Dalvik relies on the operating system for process isolation, memory management, and threading control. At compile time, a tool named dx, which is part of the Android SDK, reforms the Java Class files of the Java source, which are first compiled by a Java compiler into DEX code. All of the DEX files for a single application are then zipped

into a single Android Package file (APK) [6]. These APK files are how applications are stored on a device.

D. SECURITY

Android applications run what is known as a “sandbox,” a separate portion of the operating system without access to other system resources without the permissions of the user, given at install time. All permissions are coded when creating the manifest.xml file except one, root permission. “Rooting” a device is the process of running code on a device to allow it to attain privileged control or “root” access within the Android system, allowing ability to change operating system parameters that are normally marked read-only. On a rooted device, root user permissions are not part of the manifest file. Root permissions are obtained while running the main.java file with admin privileges (SU) and executing the Process root = Runtime.getRuntime().exec(“su”) command. Once this permission is given, the user will not be prompted for root permissions again for that specific application. As described on the Android developer page:

> A central design point of the Android security architecture is that no application, by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or e-mails), reading or writing another application's files, performing network access, keeping the device awake, etc. [4]

Due to the fact that Android sandboxes, i.e., isolates, applications from each other, applications must explicitly share resources and data. They do this by

declaring the permissions they need for additional capabilities not provided by the basic sandbox. Applications statically declare the permissions they require, and the Android system prompts the user for consent at the time the application is installed. Android has no mechanism for granting permissions dynamically (at run-time) because it complicates the user experience to the detriment of security [7].

E. MOBILE DEVICE SENSORS

Once necessary permissions have been granted, an application can access the plethora of hardware sensors installed in the mobile device using the Android APIs. The APIs use Java classes built into the Android library, to give applications the ability to use the built-in sensors to measure certain aspects of the device's environment, allowing use of them as contexts on which to allow access to other applications.

1. International Mobile Subscriber Identity

International Mobile Subscriber Identity Numbers (IMSI) were created and formatted to provide the unique international identification of mobile terminals and mobile users to enable those terminals and users to roam among public networks, which offer public network services.

The IMSI assignment guidelines and procedures are maintained by the IMSI Oversight Council (IOC). Telcordia Technologies is responsible for administering the assignment and use of IMSIs in the United States and other North American Numbering Plan (NANP) countries [8]. Telcordia Technologies, as IMSI Administrator, participates

in the management of all segments of the IMSI, but directly administers only the Home Network Identity (HNI) segment. HNIs are assignable to operators of public mobility services with international roaming capabilities. The HNI uniquely identifies the home network of a public mobility service subscriber and contains the Mobile Country Code (MCC) and the Mobile Network Code (MNC). The remaining segment of the IMSI, the Mobile Station Identification Number (MSIN), is directly administered by the network operator to which the HNI is assigned [8]. Figure 2 depicts the IMSI breakdown.

IMSI: 310150123456789

MCC	310	USA
MNC	150	AT&T Mobility
MSIN	123456789	

Figure 2. IMSI Breakdown

The MSIN is the portion of the IMSI that is the unique identifier used in the context check portion of the application. The IMSI itself is coded onto the subscriber identity module (SIM) card, which also stores security authentication and ciphering information used by the network providers to authenticate the user to the network [9]. The SIM card also stores the K_i, a 128-bit value used in authenticating the SIM to the mobile network [10]. The SIM card is specifically designed not to allow the retrieval of the K_i, using the standard card interface used by the mobile device itself, making access to the K_i

impossible without access to the carrier’s databases and a poor choice for a context check.

2. Cellular Identification Number

A Cellular Identification number (CID) is a geographically unique number used to identify a Base Transceiver Station (BTS) or a sector of a BTS within a Location Area Code (LAC) [11]. A BTS is a piece of equipment used in GSM that facilitates communication between the user equipment (i.e., mobile device) and a service provider’s network. A LAC is a set of base stations that are grouped together in order to optimize signaling. A Base Station Controller (BSC) is the system that handles allocation of radio channels and controls handovers of the device from base station to base station. When the mobile device initially connects to a cell tower, five numerical codes that represent the network to which the device is connecting are exchanged in the handshake; the state of the tower to which device is connected, the mobile country code, the carrier’s code, the location area code and the CID [11]. The device uses the CID to assist the GPS system (if the device is A-GPS capable) to narrow down its location prior to getting satellite signals. The geographical uniqueness of the CID makes it a good simple choice for use as a context in the application.

3. Global Positioning System

Global positioning system, better known as GPS, is a satellite-based, all-weather navigation system. GPS utilizing a satellite constellation of 24 active and 4 spare satellites strategically place in precise

geosynchronous orbit all around Earth. The system became operational in 1994, under the original title NAVSTAR GPS, under control of the United States Air Force [12].

Essentially, for GPS to work, the mobile device's GPS receiver must locate four or more of the satellites in the constellation and calculate the distances to each to deduce its precise location. This process is known as Trilateration. To be able to perform this calculation, the GPS receiver must know two things; the locations of at least three of the satellites it can see and the distances to the satellites. The GPS receiver deduces this information by analyzing high frequency radio signals emitted by the satellites themselves. Using precise timing measurements, the device can calculate the distance to each satellite by analyzing how long it took for the emitted signal to travel to the device from the satellite, known as propagation time. This timing is calculated using what is known as a pseudo-random code. At a previously determined time, set by system designers, both the satellite and GPS receivers begin running the same digital pattern dictated by the code. The satellite transmits this signal precisely as it is run.

Upon receipt of the satellites' signals, the GPS receiver compares the lag between its own code and that in the received signals. The length of delays between the signals is the signals' travel time. Multiplying the delay time by the speed of light, 3X10^8 m/s, results in the distance traveled by the signal. Along with its identifying information, GPS satellites transmit data containing its current location and current time, as

measure by the onboard atomic clock. Signals from the satellites all broadcast on the same frequencies; using Code Division Multiple Access (CDMA) allows messages from individual satellites to be distinguished from others using unique encodings for each satellite (the receiver must be aware of these encodings in order to distinguish the satellite signals). Once distances to the four satellites are calculated and the locations of the satellites are known, the intersection of all four distances will result in one point indicating the receiver's location

Due to the inability to spoof or fake a GPS signature (with built in anti-spoofing in the signal transmitted), it was a logical context on which to base access. Depending on the device, the overall accuracy differences using GPS are significant. With the circa 2000 removal of Selective Availability, the intentional insertion of timing errors into the GPS system to limit accuracy of non-military GPS receivers, one can expect accuracies on the order of 10 meters, depending on the chipset installed in the device.

4. Application Program Interfaces

Through the use of the LocationManager and TelephonyManager classes built into the Android Java SDK, a programmer can access a device's IMSI, CID and GPS location. These services allow applications to check for periodic updates to the system's location based on Global Positioning System measurements, cellular identification number, and/or a wireless router to which the device is currently connected, utilizing the built-in device hardware available [13]. Using a background service, these

parameters can be periodically checked allowing a programmer to base contexts on them [18].

F. SUMMARY

This chapter discussed the concept of Bring Your Own Device policies, the advantages/disadvantages of those policies, the Android operating system and its advantages/disadvantages and mobile device sensors and capabilities. It is meant to provide an understanding for an understanding of concepts used in the architectural design and prototype implementation described in the following chapters.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ARCHITECTURE

A. INTRODUCTION

This chapter presents an overview of the architecture used to design a context-based application to include background services, context checks, user authentication and the secure enclave. It describes how the contexts themselves are designed to include which contexts were chosen, and why, and the design of the enclave and its protection mechanisms.

B. APPLICATION CONTEXT DESIGN

The end-state goal of this thesis is to create a service capable of securing applications on a mobile device based on the contexts setup and controlled by the service. Using the built-in hardware common to most mobile devices and the Android SDK, it is possible to utilize the hardware as locational/user verification checks in the application. Once the contexts have been verified, the user can be allowed to run the secured applications (enclave). Figure 3 depicts a use case for the context check portion of the application.

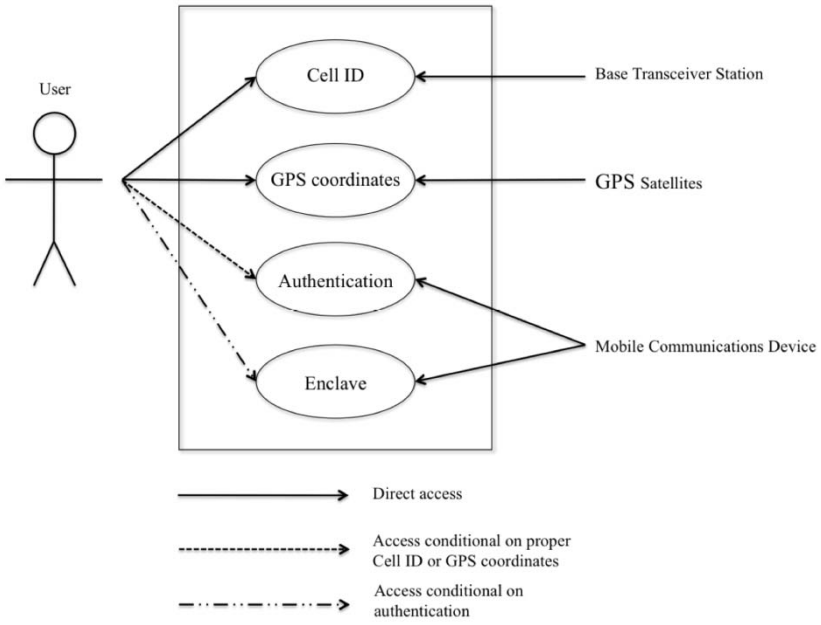

Figure 3. Context Check Use Case

1. Location Verification Context

The purpose of this context is to verify a user's location. An administrator or chaperone of restricted information may not want anyone to access the information if they are physically located outside of a specific location. An example of this might be an emergency responder located within a disaster area having access to emergency medical data about certain persons residing within the disaster area. The advantage of using location as a context, is that most mobile devices have multiple ways of determining and verifying their location (e.g., Base Transceiver Station (BTS) to which the device is currently associated, wireless access point to which the device is currently connected and GPS).

a. Cellular Identification Number

Identification of the BTS to which the device is associated can be used as a location identifier in certain contexts, as long as the BTS is stationary. Since each BTS identifier (CID) is geographically unique, using a combination of the most recent BTSs to which the mobile device has connected, a fairly reliable location can be determined. Once the CID is obtained, the application needs only to reference a table of approved CIDs to determine if the context is met. If the CID to which the device is associated is in the table of approved CIDs, the context is met. Pairing a CID context check with a more secure context is necessary due to the ability to spoof CIDs.

b. Global Positioning System

A mobile devices geographical location can be a strong and secure context on which to base access to the enclave. Similar to CID, the mobile device will calculate its geographical location using the built-in GPS system, but instead of comparing the coordinates to a preset list of approved coordinates, another calculation is done. The application takes the approved coordinates and calculates a radius around those coordinates, essentially creating a GPS fence. If the device’s measured GPS coordinates fall within the calculated GPS fence coordinates, the context is met. To prevent unauthorized interference with the GPS system, the system employs a method of encryption to prevent spoofing. (A spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an

illegitimate advantage.) Embedded within the transmitted signals from the GPS satellites, lies the P-Code (precision code), an encrypted Pseudorandom Number (PRN) unique to each satellite to prevent spoofing. This difficulty in spoofing a GPS signal makes it one of the strongest contexts on which to base access to the enclave. The following pseudo-code shows a sample calculation (Figure 4).

```
Get OFFSET
Get deviceLatitude, deviceLongitude
For GPScoord in approvedGPSTable:
    If GPScoord.latitude + OFFSET > deviceLatitude
    AND GPScoord.latitude - OFFSET < device.latitude
        If GPScoord.longitude + OFFSET > deviceLongitude
        AND GPScoord.longitude - OFFSET < device.longitude
            Grant Access
        Else
            Deny Access
    Else
        Deny Access
```

Figure 4. Context Pseudo-code

Once location has been verified for context checking, verification of the user must be accomplished prior to granting access to the secure enclave.

2. User Verification Context

The purpose of this context check is to verify the identity of the user attempting to access the secure enclave. Users can be authenticated in several different ways, the most common being challenge-response. In this application, challenge-response protocol was chosen for administrator access and IMSI was chosen for user verification.

a. Challenge-Response

Challenge-response authentication is a protocol in which one is presented with a question (challenge) and must respond with a valid answer (response) to be authenticated. The simplest example of challenge-response is password authentication, where the challenge is the password prompt and the response is the password.

b. International Mobile Subscriber Identity

As discussed earlier, the International Mobile Subscriber Identity (IMSI) is a unique identifying number associated with GSM and LTE mobile phone users. The providers use this number as a unique identifier, stored on the Subscriber Identity Module (SIM) card, installed in the device, to verify the users identity with the system as opposed to an International Mobile Equipment Identifier number, which identifies a specific mobile device. This application will use the IMSI as a verification of the user's identity as one of the contexts. Similar to CID, the application will have a secure database that stores the approved IMSIs, controlled by the system administrator, which will allow access to the applications stored within the enclave. Along with IMSI, a policy requiring the user to lock their mobile devices with a complex password/pin helps verify the identity of the user and strengthens the security of the device.

C. ENCLAVE DESIGN

The enclave will consist of a list of applications deemed to need special access controls. These applications will be fully installed on the device, but the user will

not have the ability to run them until they meet the selected access controls (contexts).

The primary means of preventing access to the applications residing within the enclave is a capability of interrupting the launch process (intent) of the restricted application. An Intent is an abstract description of an operation to be performed, such as starting an activity. When an application is launched, an intent is fired for the main activity within the application being launched [14]. Recognition of this launch process, when properly identified, allows it to be paused in order to query for required context checks.

The administrator chooses the contexts that are required to open a restricted application during the system setup in the admin settings of the enclave application. Here, the administrator can decide the contexts for which he/she would like to check and the settings for those contexts.

The design of the context checker is what enables the enclave to be properly protected. A service that runs in the background of the operating system periodically checks the last known GPS coordinates, monitors currently associated CID, and watches for firing of intents in order to identify when the user has launched an protected application.

When a user attempts to launch a protected application the contexts that are required are checked individually. Based on the results of the contexts queries, the user is either granted, or denied, access to the protected applications. There is no possible way to launch any of

the applications under the enclave’s protection via the user interfaces, without meeting the contexts. To ensure proper security, the contexts to enter an enclaved application are checked each and every time any protected application is launched/re-launched.

D. SUMMARY

This chapter provided a high-level view of the architecture of an application that securely restricts access to other applications using hardware sensor readings of the device as contexts on which to base access. The major portions of the design are the application context design and the design of the enclave itself. The next chapter will discuss the actual implementation of this architectural design into a working Android application.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. IMPLEMENTATION

A. INTRODUCTION

This chapter explains in detail the implementation of the architecture described in Chapter III of this thesis. It includes coding methodology descriptions and walkthroughs as the Administrator and a user each of which meets or does not meet the required contexts to open a protected application. The application was written for rooted Android devices, firmware 1.6 and up, and consists of 20 classes and over 2,000 lines of Java code. Root permissions allow the application admin privileges allowing it to access LogCat, to monitor for protected application launches. The enclave application was installed and tested on a Samsung Galaxy Note running Android OS 2.3.6 (codenamed Gingerbread) with root permissions.

The following sections discuss our Android application that implements the architectural design discussed in Chapter III of this thesis, hereafter referred to as the enclave application. The enclave application implements all of the features of the architectural design.

1. Key Features

- Administrator Login with challenge/response user verification.
- Administrator capability to select mobile device’s sensors to use as contexts.
- Administrator capability to modify contextual requirements to launch enclaved applications.
- Administrator capability to lock ProgramManager to prevent force-quit or uninstall of the enclave application.

- Ability to stop the launch of an enclaved application unless the administrator-selected contexts are satisfied.

B. ADMINISTRATOR

This section will describe system setup by an administrator or person controlling the access to the restricted information/application. It will discuss initial install and setup of the application, security settings and how to include applications in the enclave, to include an explanation as to how the code accomplishes this.

1. Application Installation

The application will be provided to the administrator as an APK (Android Package File), ready to install on the desired device. During installation, the administrator will be prompted to accept the user permissions required on the device. The permissions required to run the application are as follows: Fine GPS Location, Network Communication, Read Sensitive Log Data, and Read Phone Status and ID, system tools and root. Once installation is complete, the application will place a launcher icon in the Android application drawer labeled Latchkey App Protector. Figure 5 shows the installation and root permissions screens.

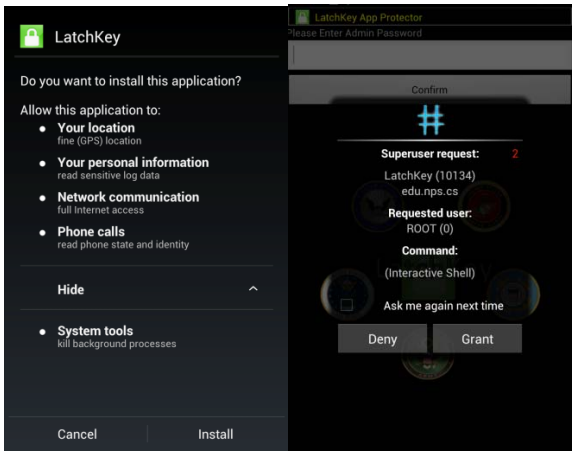

Figure 5. Application Installation Screen

a. Permissions

The various permissions necessary to install and run the enclave application enable the program to access the features and perform the functions required to meet architectural specifications. The Fine GPS Location and Network Communication permissions are required to allow the application to query the device for the necessary sensor information to determine the phone's location based on GPS and CID, respectively. The Read Phone State and ID permissions are required to allow the enclave application to query for the IMSI associated with the installed SIM card [15]. Root permissions are required to allow the application to read phone logs and pause/kill the launch activities of applications other than itself.

2. Admin Settings

After launching the application, root permissions are required prior to continuing. If the enclave application is installed on a non-rooted device, the program fails to attain root permissions at this point, resulting in the application force-quitting. The root permissions are

required to read the LogCat to monitor for application launches. Without access to LogCat, the enclave service will not function as designed. If root permissions are granted, the enclave application is launched and the administrator is prompted for an admin password, defaulted to 1234 after installation. Once the correct password is entered, the administrator can now use the context menus within the application to configure the security settings and select the applications to be protected. Figure 6 shows the application configuration screens.

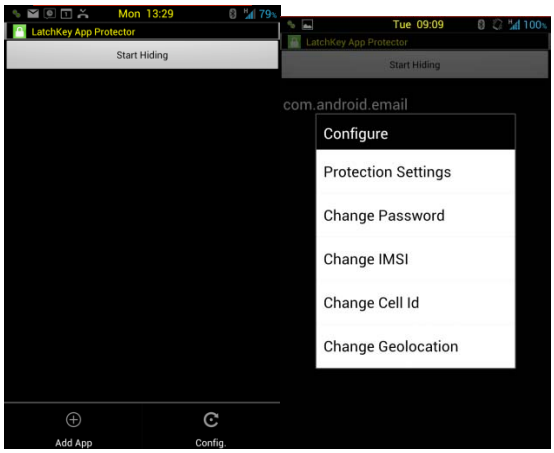

Figure 6. Application Configuration Screens

a. Protection Settings

In the protection settings menu, the administrator can select the contexts on which the enclave access is to be based (i.e., IMSI, CID, and/or GPS). This was done using a simple Boolean indicating whether the context was selected or not. Upon selecting “Save Settings” the selected context settings are saved using the SharedPreferences class built into the Android SDK. The SharedPreferences class provides a general framework that allows you to save and retrieve persistent key-value pairs

of primitive data types [16]. SharedPreferences can be used to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is “killed”). Figure 7 depicts the protection selection screen.

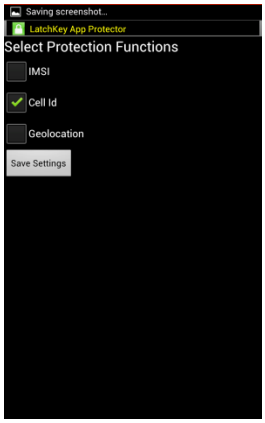

Figure 7. Application Protection Selection Screen

b. Change Password

The Change Password selection in the configuration menu simply allows the administrator to change the admin password, given that the old password is known. The stored passwords are also stored in SharedPreferences. It is important to note that a compromise of the administrator password will compromise the entire enclave application. Figure 8 is the password change screen.

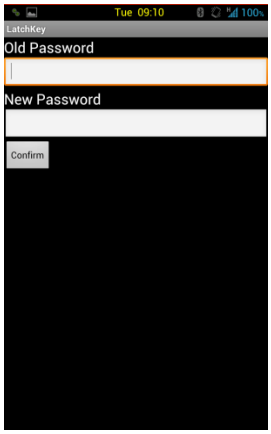

Figure 8. Application Password Change Screen

c. Change IMSI

The Change IMSI selection in the configuration menu allows the administrator to change the IMSIs on which to base access to the protected applications. Here, using a context menu, the administrator can add and delete IMSIs that are saved in SharedPreferences. As discussed in previous chapters, the IMSI is simply a unique identifier associated with the SIM card installed on the device, issued by the service provider, which can be spoofed and/or duplicated, making it a poor context check if not used in conjunction with other context parameters. Figure 9 depicts the IMSI options screens.

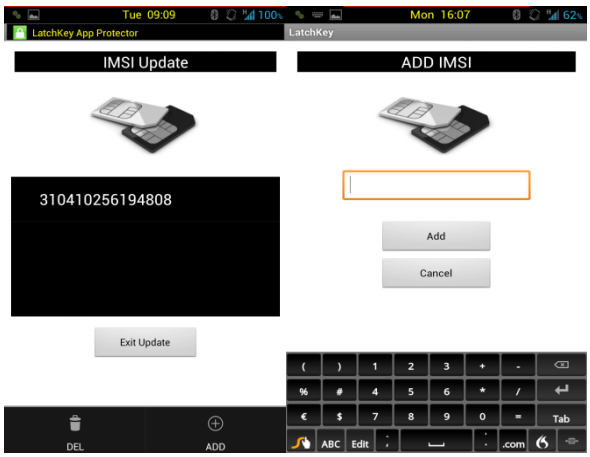

Figure 9. Application Change IMSI Screen

d. Change CellID

The Change CellID selection from the configuration menu allows the administrator to change the Cellular Identification numbers on which to base access to the protected applications. Like the Change IMSI option, the administrator can add and delete CIDs using the context menu, which is then stored in SharedPreferences. As with IMSIs, CIDs can be spoofed and/or duplicated making it a weak security context check on its own. It is also important to note that a mobile device can jump between several cellular towers even while stationary making it necessary to add all CIDs geographically located in proximity to each other, unless a specific CID is required. Figure 10 depicts the change CID screens.

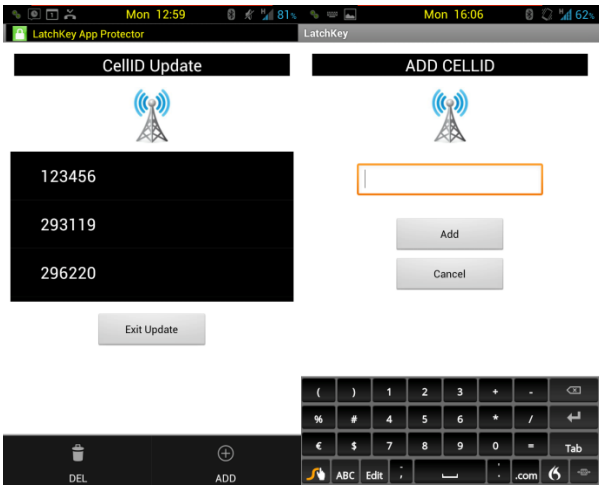

Figure 10. Application Change CID Screen

e. Change Geolocation

The Change Geolocation selection from the configuration menu allows the administrator to change the GPS coordinates that the enclave application uses to build the GPS fence on which to base access to the protected applications. Changing the GPS coordinates only changes the center point of the GPS fence used in the geolocation context and not the size of the fence itself, which is a fixed five nautical mile value. The GPS context check, done during the launch of an enclaved application, verifies that the device is geographically located within five nautical miles of the administrator's set GPS coordinates. It is possible to set several different GPS locations that are in proximity to each other to create a larger GPS fence. The GPS options screens are shown in Figure 11.

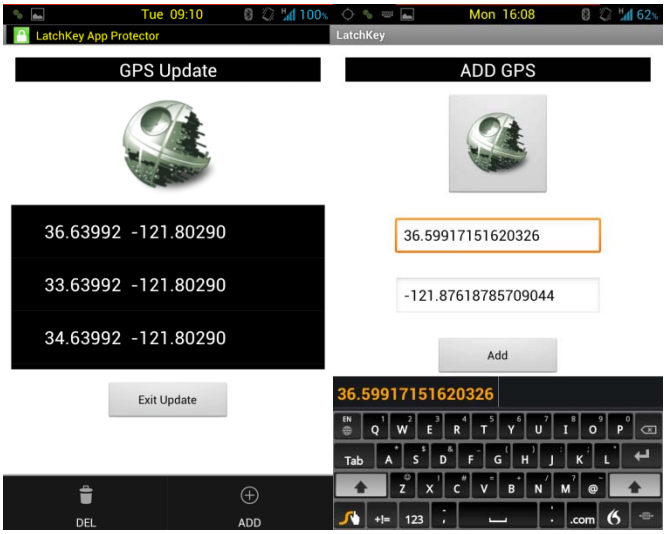

Figure 11. Application Change GPS Screen

3. Add App

The Add App selection from the context menu allows the administrator to select the applications to protect (i.e., place under enclave protection). Once Add App is selected, the enclave application provides the administrator with a list of all Main activities stored on the device. A Main activity is the activity within an application that allows it to launch in a Dalvik VM [17]. Selecting a launch activity, within this menu, adds it to a list stored in the enclave applications SharedPreferences. It is only possible to enter an application on the Android OS via the applications associated Main activity, delineated in the applications associated manifest file. Upon exiting this menu, the administrator is presented with a list of the application launch activities that are selected and stored in SharedPreferences, which will be used later in the background service.

It is important to note that selecting applications that are to be locked are not the only selections that must

be made on this step of the configuration. The administrator must also lock any applications or system settings that provide for stopping other applications or uninstalling packages (i.e., program manager) [4] in order to keep a user from force-quitting or uninstalling the enclave application itself. This requires an in depth knowledge of the launching applications and the system settings of the device on which the enclave application is to be installed. Failure to lock these features would allow a malicious user to force-quit or uninstall the enclave application thereby completely removing its security features.

4. Start Hiding

Once the administrator has configured all of the security settings for the enclave and selected the applications that are to be included in the enclave, he/she must click the Start Hiding button to enable the security features and “lock the enclave.” Once the Start Hiding button is clicked, the enclave application starts a background service that monitors the Android LogCat log. Android LogCat contains logs from various applications and portions of the system that are collected in a series of circular buffers, which then can be viewed and filtered by the logcat command. LogCat can be viewed from an ADB shell to view the log messages. ADB (Android Debug Bridge) is a tool that comes with the Android SDK that allows you to control and interface with your Android device [18].

The enclave application uses this method to monitor the LogCat log for application launches. The background service is essentially looking for activity launch events

that correspond to the launch activities that were selected by the administrator during the Add App portion of the enclave application configuration. When an activity launch is detected, the background service checks to see if the activity being launched matches one selected by the administrator to be protected.

C. USER

The intended user of the enclave application is a person who requires access to restricted applications/ information when a preselected set of contexts is met. The enclave application can be preinstalled on an issued device or directly installed on a user's personal device and configured by a system administrator. Even when installed on the user's personal device, the administrator privileges of the service are protected with a password. When the user attempts to launch a restricted application one of two things will occur: the contexts are satisfied and the application launches or the contexts are not satisfied and the launch of the application is blocked.

1. Launching a Protected Application

Upon attempting to launch a protected application, the background service, running since the administrator configured the enclave application, senses the launch via LogCat. If the activity name matches one of the selected activities to be protected, the activity of the protected application is then halted and the context checking activity of the enclave application is started.

a. Contexts Satisfied

The context checking activity of the enclave application checks to verify that the contexts configured by the administrator during setup are met. IMSI, CID and/or GPS coordinates are queried and checked to see if they match the values stored in SharedPreferences. If the contexts are met, the user is shown that the contexts are satisfied and they are given the option of clicking continue, thereby un-pausing the restricted application and allowing it to run as usual. An “Access Granted” screen is shown in Figure 12.

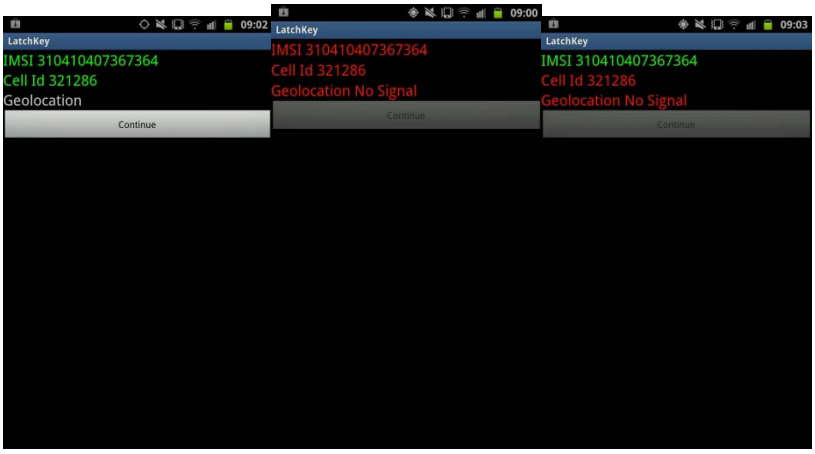

Figure 12. Context Monitor Screen

b. Contexts Not Met

As above, the context checking activity of the restricted application, upon launch, will attempt to verify if the contexts required to access a protected application are satisfied. If any of the contexts set by the administrator are not satisfied, the launching activity of the restricted application is killed and the user is not allowed to continue until the contexts are met. The access

denied screen is shown in Figure 12. It is important to note that if the user navigates away from an application that is under protection, the contexts will be checked once again upon returning, and in the event they are not met, the user will not be allowed back into the application and all unsaved data will be lost.

D. SUMMARY

In summary, the enclave application prototype satisfies all of the architectural design requirements presented in Chapter III of this thesis. The enclave application has the capability to protect and restrict the launch of selected applications on an Android device based on selected contexts using the hardware sensors built into the mobile device itself.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

This thesis has developed an advanced Android service extending the ideas presented in a thesis written by Kevin J. LaFrenier [2] in September 2011. The idea was to create a method of protecting restricted applications on an Android device from compromise by constraining the applications' launch based on a set of contexts over which the user has no control.

We used the Android Java SDK and its associated libraries in the Eclipse IDE (Integrated Development Environment) to build a prototype application that prevented the launch of other, preselected applications, based on a set of contexts measured by the mobile device's hardware.

The first portion of the thesis defined the contexts on which to base access to the protected applications. The first context used in the service was the physical location of the mobile device. The enclave service uses two means of determining location: Cellular Identification Number assigned to the Base Transceiver Station and Global Positioning System Coordinates. Both of these values are measured using the built-in Android LocationManager class utilizing the mobile device's hardware. Once these values are obtained, they are crosschecked with the list of approved CIDs and GPS coordinates; the list of approved values being configured on installation by the system administrator to either allow or no to allow access to the sensitive applications, also selected by the administrator.

The second context coded into the service is the International Mobile Subscriber Identity (IMSI). Since IMSIs are attached to user accounts, it was chosen as a pseudo-verification of the user and/or device. The TelephonyManager class of the Android SDK is used to query for the IMSI assigned to the SIM card installed on the device. This measured IMSI is then crosschecked with a list of approved IMSIs to determine whether or not to allow access to a protected application.

Based on the measured contexts, once met, the user is allowed to launch any of the protected applications. If one or more of the selected contexts are not met, the enclave service uses the Android OS to block the launch of the application, until said contexts are satisfied. Once set-up, the enclave service successfully restricts launching an application based on a set of contexts utilizing hardware already installed on the device. Since the contexts are only checked at launch time, there is no protection if any of the contexts fails after the user has launched an application.

B. FUTURE WORK

The design and creation of this prototype enclave service was intended to show that it is possible to build a service that successfully restricts access to other applications based on a set of contexts. There are still several areas requiring further research prior to deployment. These areas focus on user identification, multi-level contexts, remote context changes, and user termination resistance.

1. User Identification

In creating the enclave application, the idea of user identification was ignored in order to focus on designing a protection scheme reliant on the mobile device's sensors as contexts. In a real-world deployment of a security service, the users themselves must be verified prior to allowing access to restricted applications. The addition of a context employing a challenge/response protocol would serve as a reliable means of user identification prior to using the mobile device's resources to unnecessarily check contexts in the event the user is unauthorized.

2. Multi-level Contexts

A single device might contain differing levels of protected applications. In this situation, an enclave service that allows for the use of differing contexts based on the required protection level of the application the user is attempting to access will prevent an over-protection of certain applications. As a result, a single device might contain multiple applications at differing protection levels, each able to be accessed at different moments in time/location by different users.

A more complicated, tiered-context check would also make the enclave service more resilient to compromise. An off-device context check, requiring the presence of other devices and/or users would provide this increased protection. A simple presence check of other devices in the area in order to access restricted operations would prevent a compromised device, which may have fallen into malicious hands, from accessing restricted resources, thereby mitigating the compromise of the device.

Establishing a means of monitoring contexts while the user is accessing a protected application so that it can be killed if any of them fail at any time would provide an increased layer of protection. This service would provide real-time protection to the enclaved applications.

3. Remote Context Changes

In the event that a device is lost or stolen, there is currently no means of keeping a malicious user from accessing the applications protected by the enclave service if the contexts are met. A periodic means of updating the contexts stored in the SharedPreferences of the enclave application would provide an administrator or information manager the ability to remove contexts, essentially locking a device out of the protected applications. The ability to add or change contexts remotely allows an administrator to grant/remove a device user’s access to a protected application, to which the user would not otherwise have had access.

4. User Termination Resistance

In the prototype enclave service, there are no standard protections keeping a user from self-terminating the enclave service. The administrator, at setup time, must ensure that any application with force-stop and uninstall capabilities (i.e., Android PackageManager) are also under the enclave application’s protection. This ensures that the contexts set to protect the applications within the enclave are met prior to allowing a user to be able to uninstall/force-quit any of those applications.

There are currently no automatic protections for keeping the enclave application from being force-quit or uninstalled by the user.

In the event of a full mobile device reboot, the enclave service is not included in the boot procedure, therefore, it will not start and thus leaving the applications, previously protected based on context checks prior to the reboot, vulnerable. The simple addition of an intent filter monitoring for the BOOT_COMPLETE action can start the background service in the event of a device reboot, making the enclave application more resilient to user termination.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] GSM World, “History,” [Online]. Available: http://www.gsmworld.com/about- us/history.htm.

[2] K. LaFrenier “Mobile security enclaves.” M.S. thesis, Computer Science, Naval Postgraduate School, California, 2011.

[3] Android, “What is Android?” [Online]. Available: http://developer.android.com/guide/basics/what-is-android.html.

[4] Android. “Security architecture.” [Online]. Available: http://developer.android.com/guide/topics/security/permissions.html.

[5] Dalvik Virtual Machine, “Introduction.” [Online]. Available: http://www.dalvikvm.com.

[6] Dalvik Virtual Machine, “DEX file format.” [Online]. Available: http://www.dalvikvm.com.

[7] J. Six, “Android architecture.” In *Application Security for the Android Platform*, Sebastopol, CA, O’Rielly Media 2011, pp 13–24.

[8] Wikipedia. “International mobile subscriber identity number,” [Online]. Available: http://en.wikipedia.org/wiki/International_Mobile_Subscriber_Identity.

[9]]K. Vedder, “GSM: Security, services and the SIM,” *Computer Science*, 1528, pp. 233, 1998.

[10] ETSI, “Digital cellular telecommunications systems; specifications of the Subscriber Identity Module – Mobile Equipment Interface (SIM-ME), “GSM 11.11, pp.33, 1995.

[11] Wikipedia, “Cell ID,” [Online]. Available: http://en.wikipedia.org/wiki/Cell_ID.

[12] Space and Tech, “NAVSTAR GPS – Summary.” [Online]. Available: http://www.spaceandtech.com/spacedata/constellations/navstar-gps_consum.shtml.

[13] Android, “Package android.Location,” [Online]. Available: http://developer.android.com/references/android/location/package-summary.html.

[14] Android,”Intents and intent filters,” [Online]. Available: http://developer.android.com/guide/topics/intents/intents-filters.html.

[15] Android, “Public static final class Manifest.permission,” [Online]. Available: http://developer.android.com/references/android/Manifest.permission.html.

[16] J. Six, “Application permissions.” In *Application Security for the Android Platform*, Sebastopol, CA: O’Rielly Media 2011, pp. 25–41.

[17] Android,”Activities,” [Online]. Available: http://developer.android.com/guide/topics/fundamentals/activities.html.

[18] Android, “Background services,” January, 2012 http://developer.android.com/guide/components/services.html.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Gurminder Singh
   Naval Postgraduate School
   Monterey California

4. Mr. John Gibson
   Naval Postgraduate School
   Monterey, California